Image Classification Capstone Project

In this project we are going to classify numbers from the dataset MNIST by using Pytorch and Keras. And we are going to compare the both methods

Image classification refers to a process in computer vision that can classify an image according to its visual content. We will use the popular MNIST dataset, a dataset of images, for a change.

The MNIST database, short for Modified National Institute of Standards and Technology database, is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning.

The MNIST database contains 60,000 training images and 10,000 testing images of digits written by high school students and employees of the United States Census Bureau.

We will get to compare how conventional neural networks in Keras compare to convolutional neural in Pytorch and Keras.

Layers in a CNN

We are capable of using many different layers in a convolutional neural network. However, convolution, pooling, and fully connect layers are the most important ones.

Convolutional Layers

Convolutional layer is the very first layer where we extract features from the images in our datasets. Due to the fact that pixels are only related with the adjacent and close pixels, convolution allows us to preserve the relationship between different parts of an image. Convolution is basically filtering the image with a smaller pixel filter to decrease the size of the image without loosing the relationship between pixels.

Keras

Reshaping and Normalizing the Images

To be able to use the dataset in Keras API, we need 4-dims numpy arrays. However, as we see above, our array is 3-dims. In addition, we must normalize our data as it is always required in neural network models. We can achieve this by dividing the RGB codes to 255 (which is the maximum RGB code minus the minimum RGB code).

Building the Convolutional Neural Network

We will build our model by using high level Keras API which uses either TensorFlow or Theano on the backend. I would like to mention that there are several high level TensorFlow APIs such as Layers, Keras, and Estimators which helps us create neural networks with high level knowledge. However, this may lead to confusion since they all varies in their implementation structure.

Compiling and Fitting the Model

With are going to create an non-optimized CNN. Wich later we optimizer with a given loss function which uses a metric. Then, we can fit the model by using our train data.

Valuating the Model

Finally, you may evaluate the trained model with x_test and y_test using Pytorch and Keras with CNN and we compare both models.

Results

By using Pytorch and CNN we obtain the following results

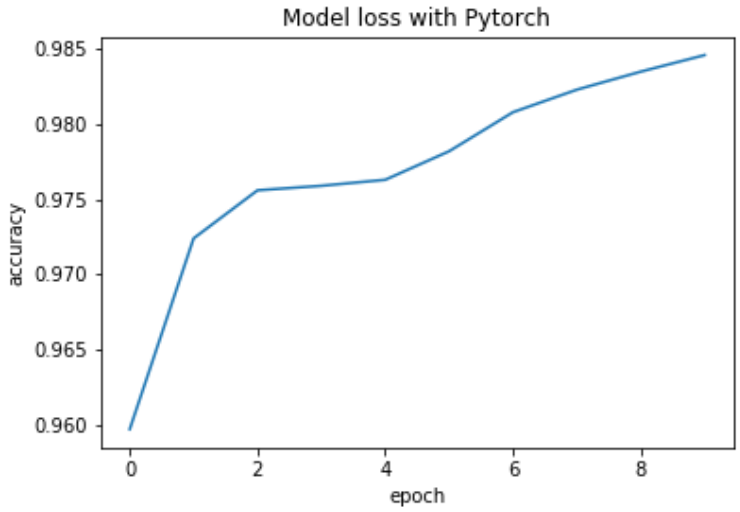

Figure 1.0 Accuracy of the model in Pytorch by using CNN

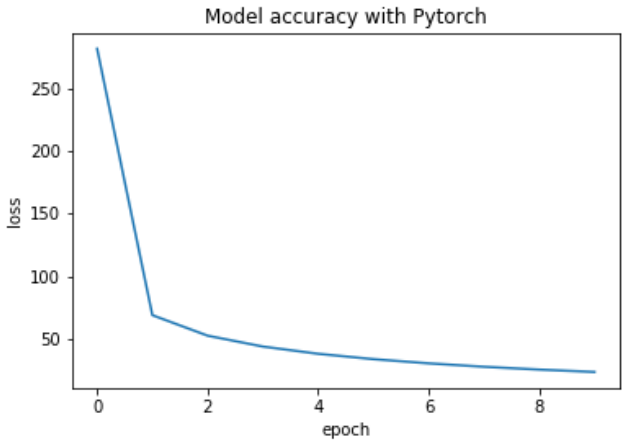

Figure 2.0 Loss of the model in Pytorch by using CNN

The Accuracy is around 0.975 with 4 epochs. Repeating the same procedure but now by using Keras without CNN we obtain an Accuracy of 0.979% and of Error: 0.021

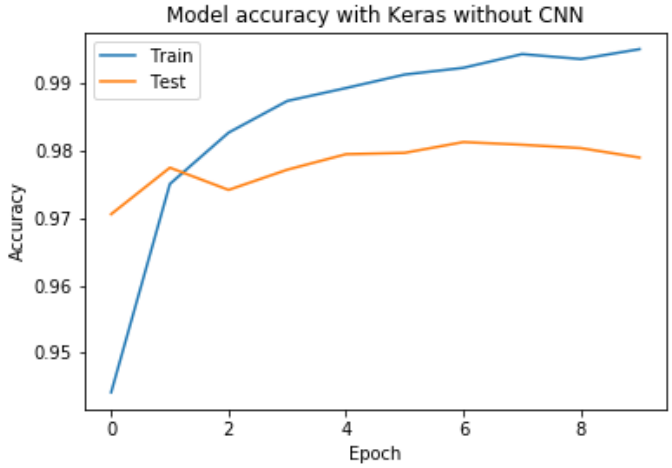

Figure 3.0 Accuracy of the model in Keras without CNN

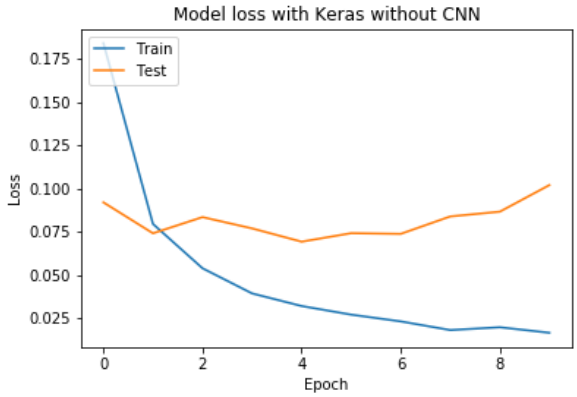

Figure 4.0 Loss of the model in Keras without CNN

We have found that the accuracy curve in Pytorch from Fig 1.0 and Keras without CNN in Fig. 3.0 for the train set has convex shape.

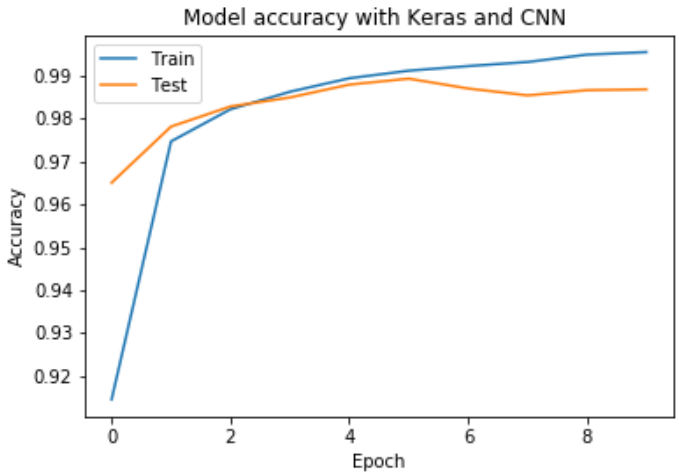

Figure 5.0 Accuracy of the model in Keras by using CNN

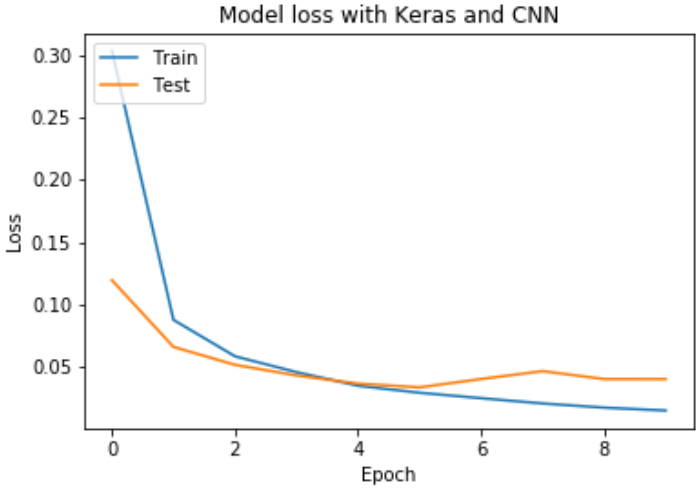

Figure 6.0 Loss of the model in Keras by using CNN

Meanwhile by using the CNN in Keras the results gives an accuracy: 0.9869 and Error: 1.31.

We have noticed that the training part of the model by using the Pytorch scheme requires more computing power than in Keras with this dataset. So by increasing the set of data the convolution procedure might be a problem in terms in time computing. In this sense Keras without covolution provides faster training times and acceptable accuracy values within few epochs.

Conclusion

For such kind of classification of The MNIST database which contains a training set of 60,000 examples, and a test set of 10,000 examples, the Keras scheme without CNN provides a faster training model with an Accuracy of 0.979% and of Error: 0.021 meanwhile the Pytorch scheme with CNN provides a bit lower Accuracy around 0.975% with more time computing during the training step.